Assessing performance of prediction rules in machine learning

*Rory Martin[1†] &
Kai Yu[2,3]*

*[†]Author for correspondence
[1]Millennium Pharmaceuticals, Cambridge MA 02139, USA
E-mail: rory.martin.phd@gmail.com
[2]Washington University, St. Louis, MO 63110, USA
[3]Current address: National Cancer Institute, Division of Cancer Epidemiology and Genetics, Bethesda MD 20892, USA*

Introduction: An important goal in machine learning is to assess the degree to which prediction rules are robust and replicable, since these rules are used for decision making and for planning follow-up studies. This requires an estimate of a prediction rule's true error rate, a statistic that can be estimated by resampling data. However, there are many possible approaches depending upon whether we draw observations with or without replacement, or sample once, repeatedly, or not at all, and the pros and cons of each are often unclear. This study illustrates and compares different methods for estimating true error with the aim of providing practical guidance to users of machine learning techniques. **Methods**: We conducted Monte Carlo simulation studies using four different error estimators: bootstrap, split sample, resubstitution and a direct estimate of true error. Here, 'split sample' refers to a single random partition of the data into a pair of training and test samples, a popular scheme. We used stochastic gradient boosting as a learning algorithm, and considered data from two studies for which the underlying data mechanism was known to be complex: a library of 6000 tripeptide substrates collected for analysis of proteasome inhibition as part of anticancer drug design, and a cardiovascular study involving 600 subjects receiving antiplatelet treatment for acute coronary syndrome. **Results**: There were important differences in the performance of the various error estimators examined. Error estimators for split sample and resubstitution, while being the most transparent in action and the simplest to apply, did not quantify the performance of prediction rules as accurately as the bootstrap. This was true for both types of study data, despite their highly different nature. **Conclusions**: The robustness and reliability of decisions based on analysis of genomics data could, in many cases, be improved by following best practices for prediction error estimation. For this, techniques such as bootstrap should be considered.

Keywords: bootstrap, machine learning, Monte Carlo simulation, prediction rule, split sample, stochastic gradient boosting, true error

Predictive performance has been described as a guiding principle for selecting a prediction rule [1]. Not only does it quantify our degree of belief that conclusions based on a model are robust and replicable, but it facilitates comparisons between different hypotheses (for example, interactions present/absent), different machine learning methods (for example, one tree versus an ensemble of them), and different experimental designs. Being able to reliably quantify confidence in a prediction rule is highly important since one of the potential pitfalls of machine learners is they always generate a prediction rule.

A theoretical ideal for measuring predictive performance is the true error rate [2] of the prediction rule. Given a sample S drawn from some distribution F, with inputs x and observed response y, the true error rate of a prediction rule $M_S(\,)$ fitted to S is:

(1) Expectation$\{\mathrm{Err}(y, M_S(x))\}$, over (x,y) drawn from F

where $M_S(x)$ is a predicted response, and $\mathrm{Err}(\cdot)$ is an error function comparing observed vs predicted response. Examples include 0–1 loss for binary response and squared error for continuous response [3]. Equation (1) is a conditional generalization error, conditioned upon having chosen S as the training sample [2].

In practical terms, true error is not a directly useful quantity, since the very large and independent sample it demands is rarely, if ever, available in real-world studies. However, a number of estimators are available that seek to approximate true error. These include the bootstrap, split sample and resubstitution. We now describe each method briefly, and compare their strengths and weaknesses (see [2] for theoretical details).

The resubstitution error rate (sometimes called apparent error) is formed by taking the sample S and dropping it back through the rule $M_S(\,)$ that was fitted to it. Resubstitution is highly attractive because of its economy and simplicity: 100% of the data are used for model training, and there are no tuning parameters involved constructing the error estimator. Its Achille's heel is its downwards bias as an estimator of true error, causing it to paint an overly optimistic picture of predictive performance (for more details, see [2]).

Split sample is a commonly used technique in which a single randomized partition is used to divide the sample into two parts – a training sample used to fit the rule and a test sample with which the rule's performance is quantified. In our experience, split sample has very wide appeal and usage among all types of scientists. Its appeal is threefold. First, it is quick to perform. Second, there is maximal clarity regarding the role of data: each datum is used either to train the rule or to test it (but never both). Third, and possibly most importantly, the method is perceived as impartial, since the test sample (once chosen) can be kept under lock and key.

Completing our offering of sampling methods is the bootstrap [2,4]. A point in favor of the bootstrap is its theoretical motivation, with the method described as a nonparametric maximum likelihood approach. It forms an empirical distribution – a probability distribution formed directly from a sample's data points, rather than specified as a mathematical function – by placing a probability mass of $1/n$ on each of the n observations in sample S, then uses this empirical distribution as a surrogate for the underlying, but unknown population distribution F. Observations are drawn with replacement to create a replicate sample S_i, and the process is repeated 100 or so times to form a bootstrap ensemble of such samples.

Despite its theoretical attraction and published studies establishing its performance versus other sampling methods (see for example [2,4]) the bootstrap is often viewed with suspicion. Aside from the fact it is more complicated to implement and takes much greater computational effort to run, each observation plays dual roles in a bootstrap, being in turn a member of both training and testing samples. In layman's terms, a bootstrap appears to some to be a free lunch.

Furthermore, we normally perform a finesse when constructing a bootstrap estimator of true error. For resubstitution, the final model (rule) that we choose for future prediction and for extracting knowledge about system behavior is the same model used to estimate error. In contrast, the ensemble of models created during bootstrap are fitted afresh and act as a surrogate for the final model fitted to the entire sample. Of course, the same kind of finesse is also used for the split sample approach and for other estimators, such as cross-validation.

When discussing these points with collaborators whose expertise lay in areas other than mathematics, we found ourselves advocating the use of bootstrap estimators, but facing pushback due to the perception that advantages of simpler methods such as impartiality had been lost, with an uncertain gain. Furthermore, differences between approaches were multifactorial, with it being possible to resample once, repeatedly, or not at all, to draw observations with replacement or without, and to put various degrees of effort on training a model versus testing it.

In order to try and clarify the situation, we decided to answer the following question: in situations representative of the complicated data we routinely have to study, what practical differences are there between these competing estimators? To answer this question, we designed Monte Carlo simulation experiments based on real data from two in-house studies to compare the above error estimators. The simulations were designed to estimate the distributions for the various error estimators, as opposed to single point estimates, in the hope that these distributions would be more informative.

In the following section we describe the design of the simulation and indicate how various estimators were constructed. We also provide an overview of the two studies themselves and, finally, we present our findings.

Methods

Monte Carlo simulation can be used to compare different analysis strategies in situations in which theoretical approaches are intractable. One way of applying the technique is to construct a model to generate synthetic data, but although this gives fine control over data generated, it requires explicit assumptions concerning how inputs are functionally related to response. Such assumptions can be difficult to formulate in ways that reflect behavior of real world systems.

A different approach for comparing analysis strategies is to select a range of samples of data and apply to each, in turn, the strategies under investigation [5–7]. Although this approach guarantees relevance of the data being studied (with careful choice of sample) the drawback is that only point estimates of the relevant statistics are generated.

For the current research we employ a hybrid design that attempts to leverage advantages of both approaches. Briefly, we use two different levels of sampling: the first generating an ensemble of data sets that are used as baseline samples to analyze, and a second level that is used to construct various estimators of true error. Using this design, we can use real data to compare error estimators in terms of their distributions as opposed to at specific points. A similar approach has been used by Molinaro and co-authors [8] for the study of binary response data.

We now describe our method in detail.

Simulation design

A schematic for the simulation design is shown in Figure 1. The simulation involves two levels of sampling. We start by choosing one of the two data sets described below, and use it to define a population from which all the first level sampling is performed (Figure 1, 'Sample S', purple box).

For $i = 1,...,n$, we apply a randomized partition to sample S to create a pair of samples, a baseline sample $S_{\text{Base}}(i)$ and an independent test sample $S_{\text{Test}}(i)$ (Figure 1, blue boxes). For a given i, every observation in S is in one and only one of $S_{\text{Base}}(i)$ and $S_{\text{Test}}(i)$.

To each baseline sample $S_{\text{Base}}(i)$ a prediction rule $M_{\text{Base}}(i)$ is fitted. The aim of the simulation is to compare different ways of estimating this rule's error rate. We directly estimate the true error of this rule by dropping the corresponding independent test sample $S_{\text{Test}}(i)$ through it, and denote this error as $\text{Err}_{\text{True}}(i) = \text{Err}(M_{\text{Base}}(i), S_{\text{Test}}(i))$. The rule's resubstitution error is estimated by dropping the baseline sample $S_{\text{Base}}(i)$ through it, and is denoted as $\text{Err}_{\text{Resub}}(\text{i}) = \text{Err}(M_{\text{Base}}(i), S_{\text{Base}}(i))$.

Two other error estimators are investigated here – split sample and bootstrap – and neither requires an independent test sample.

For split sample, we apply a randomized partition to sample $S_{\text{Base}}(i)$ to create a pair of subsamples $S_{\text{Split1}}(i)$ and $S_{\text{Split2}}(i)$, with the proportion of observations in $S_{\text{Split1}}(i)$ denoted by p. We fit a prediction rule $M_{\text{Split1}}(i)$ to the first sample of the pair and calculate the rule's error by dropping the second sample through it. The split sample estimate of error is denoted by $\text{Err}_{\text{Split}}(i) = \text{Err}(M_{\text{Split1}}(i), S_{\text{Split2}}(i))$.

For bootstrap, we draw an ensemble of bootstrap samples $S_{\text{Boot}}(i,j)$ for $j = 1,...,100$ by subsampling from $S_{\text{Base}}(i)$ with replacement, using a sample size equal to $S_{\text{Base}}(i)$ (Figure 1, first row of pink boxes). For each bootstrap sample $S_{\text{Boot}}(i,j)$, its corresponding 'out-of-boot' sample $S_{\text{OOB}}(i,j)$ is the set of all observations in $S_{\text{Base}}(i)$ not in $S_{\text{OOB}}(i,j)$ (Figure 1, second row of pink boxes). We fit a prediction rule $M_{\text{Boot}}(i,j)$ to bootstrap sample $S_{\text{Boot}}(i,j)$ and calculate its error by dropping $S_{\text{OOB}}(i,j)$ through it. The bootstrap estimate of error $\text{Err}(i)_{\text{Boot}}$ of rule $M_{\text{Base}}(i)$ is the mean of $\text{Err}(M_{\text{Boot}}(i,j), S_{\text{OOB}}(i,j))$ over $j = 1,...,100$.

Thus, for a given model $M_{\text{Base}}(i)$ we have a direct measure of true error $\text{Err}_{\text{True}}(i)$ plus three different types of error estimator: resubstitution error $\text{Err}_{\text{Resub}}(i)$, split sample error $\text{Err}_{\text{Split}}(i)$, and bootstrap error $\text{Err}_{\text{Boot}}(i)$. The merit of each estimator is quantified by its difference with respect to true error: $\text{Err}_{\text{True}}(i) - \text{Err}_{\text{Resub}}(i)$ for resubstitution, $\text{Err}_{\text{True}}(i) - \text{Err}_{\text{Split}}(i)$ for split sample, and $\text{Err}_{\text{True}}(i) - \text{Err}_{\text{Boot}}(i)$ for bootstrap – with the best estimators having the smallest difference. When these differences are squared and averaged, we generate the mean square error of each statistic as an estimator of true error.

Machine learner

To construct prediction rules, we chose a tree-based machine learner called stochastic gradient boosting [9]. In brief, the method parameterizes predicted outcome as an additive expansion in which the basis functions are small trees. Boosting retains all the advantages of single-tree methods save for simplicity, but tends to produce stronger predictive performance. The metaparameters we used for boosting were 100 trees in a model, shrinkage of 0.1, stochastic subsampling set at 60% of the sample, and tree size limited to two splits. These values were chosen on the basis of our past experience with this algorithm.

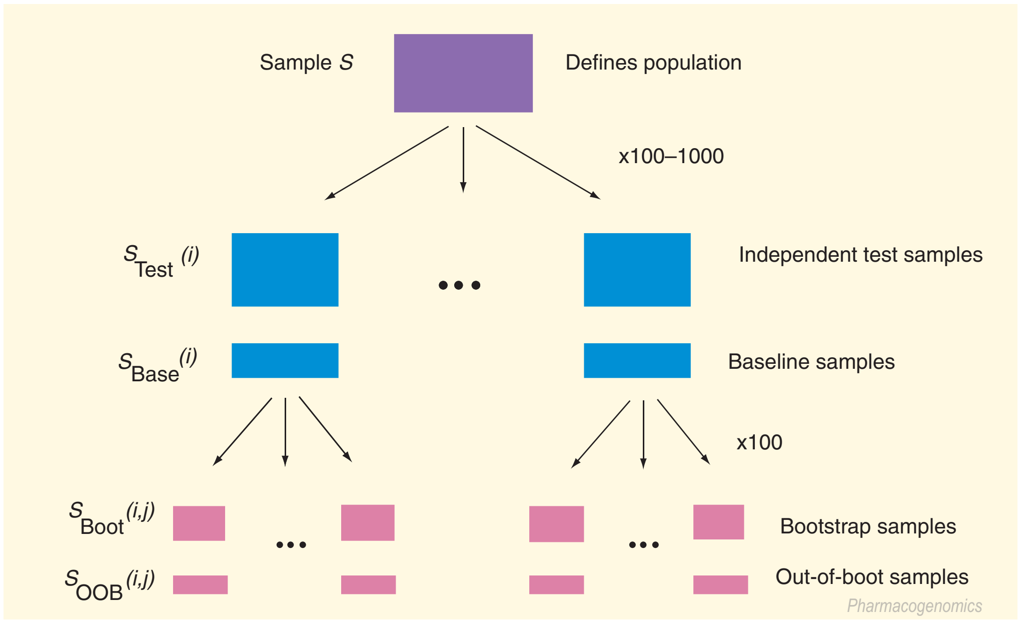
Data sets

Two data sets were used as the basis for (separate) simulation experiments. They were chosen because the underlying systems were known to be relatively complex, and as sample sizes were sufficiently large to permit the simulation design we employed.

Proteasome data

The first data set comes from a study of the proteasome, a complex of enzymes within cells that breaks down proteins that have been tagged for removal. Increased proteasome action is thought to play a role in certain types of cancer, and there is interest in studying proteasome behavior in order to design anticancer drugs that work by proteasome inhibition. Experimental data were provided in the form of a library of approximately 6000 tripeptide substrates catalyzed by the proteasome. Response is

Figure 1. Experimental design for the simulation.



We start by repeatedly sampling data from one of the listed studies (proteasome or antiplatelet) to create pairs of baseline and test samples (blue boxes). Prediction rules are fitted to baseline samples, and the error rate of each rule is estimated using either the baseline sample itself (resubstitution error), the corresponding independent test sample (direct estimate of true error), the ensemble of bootstrap samples (pink boxes), or the split-sample samples (not shown).

a quantitative readout of fluorescence that occurs when 7-amino-4-methylcoumarin (AMC) markers tagged to the substrates at one end are cleaved by the proteasome enzyme.

Each substrate was annotated using 18 molecular properties such as Van der Waal surface area, total atom count, partition coefficient, and so on. Each of these features was measured for the substrate as a whole, as well as separately for its three amino acid positions, giving a total of $4 \times 18 = 72$ inputs. The resulting proteasome data are in the form of quantitative structure–activity relationship (QSAR) data.

Antiplatelet data

The second data set is an antiplatelet survey consisting of medical records from 585 subjects admitted to hospital with a form of cardiovascular disease called non-ST elevation acute coronary syndrome. The data were collected in collaboration with MarketRx, Inc. (NJ, USA) to better understand how patient characteristics and other factors influence whether, when and how patients receive various types of therapy.

A total of 25 mostly categorical inputs were used, including patient demographics such as age, gender and weight, risk factors such as diabetes, clinical measures such as cardiac enzymes, the length of time patients waited for treatment, and whether or not patients received various kinds of antiplatelet drugs. For the simulation, we chose a binary variable for response that measured whether or not the patient received a percutaneous coronary intervention (PCI) procedure to remove blockages in coronary arteries. The proportion of patients receiving PCI was $p(\text{PCI} = \text{yes}) = 0.53$.

Results

Simulation experiments were conducted separately for proteasome and antiplatelet data. In both cases, we generated 1000 replicate baseline samples $S_{\text{Base}}(i)$ in order to construct distributions for estimators of true error, resubstitution error and split sample error. Due to computational constraints, bootstrap analysis was limited to a subset of 100 of the baseline samples, and for each of these a bootstrap ensemble of 100 replicates was generated (total of 10,000 bootstrap samples).

For split sample error estimation, we tried a range of values of the splitting proportion p in order to investigate the effect of allocating more or less data to the training sample. The values used were 0.2, 0.33, 0.5, 0.67 and 0.8.

For antiplatelet data, all sampling was stratified on the response frequency $p(\text{PCI} = \text{yes}) = 0.53$ in the original sample, meaning we defined separate empirical distributions for patients who did and did not receive a PCI to hold this frequency constant.

The following sample sizes were used. For proteasome data, 2000 observations were allocated to a baseline sample, and the remaining 4000 were put in its matching test sample. For antiplatelet data, 300 observations were used for baseline, and the remaining 285 for testing. Baseline samples much smaller than 300 tended to give relatively weak prediction rules that were not useful for error comparison.

Results for proteasome data are shown in Table 1 and Figure 2, and for antiplatelet data in Table 2 and Figure 3. The figures show distributions for error differences as defined previously, while the tables summarize these data in terms of mean square error using the usual bias-variance decomposition. This decomposition expresses the mean squared error of an estimator (relative to truth) as the sum of two terms: the estimator's variance, plus the squared distance between the mean of the estimator and the statistic being estimated. This second term is the square of the bias; for the split sample estimator the bias is:

(2) $$\text{bias}(\text{Err}_{\text{Split}}(\cdot)) = \text{mean}(\text{Err}_{\text{True}}(\cdot)) - \text{mean}(\text{Err}_{\text{Split}}(\cdot))$$

Several effects are apparent from Tables 1 and 2. First, the qualitative behavior of the group of estimators is strikingly similar for the proteasome and antiplatelet data, despite the very different nature of these two samples. Second, the resubstitution estimator is strongly biased downwards, as has been described by other authors [2], meaning it makes prediction rules appear better than they really are. Bootstrap has the smallest variance of any estimator (other than resubstitution), which it achieves by averaging estimates over many replicates. Its bias is approximately equal to split sample bias at $p = 0.67$, which is consistent with the fact that the effective size of a bootstrap sample is only $1–1/e \approx 0.63$ due to the presence of duplicate observations (sampling is with replacement). Note, however, that bootstrap standard deviation is no more than half that of split sample at this point.

For both types of data, we can quantify the effect of sample size upon the bias and variance (standard deviation) of the split sample estimator by considering bias and variance as functions of p, then using the data contained in Tables 1 and 2 to estimate rates of change with respect to this parameter. For instance, the first-order, one sided finite difference (a type of numerical derivative) for standard deviation at $p = q_1$ is:

(3) $$\left.\frac{\partial \text{SD}(p)}{\partial p}\right|_{p = q_1} = \frac{\text{SD}(q_1) - \text{SD}(q_2)}{q_1 - q_2}$$

For both the proteasome and antiplatelet data, the rate of change of the standard deviation of the split sample estimator is greatest when p is large, being over an order of magnitude greater than its rate of change when p is small. In contrast, the rate of change of the bias of this estimator is smallest when p is large, being only half its size at small p values. When p is large, the proportional change in size of the test sample is greatest while that of the training sample is smallest, and the converse is true when p is small. We conclude from these results that the dominant effect driving variance of the split sample estimator is size of the test sample, while the training sample size tends to control estimator bias.

Conclusions

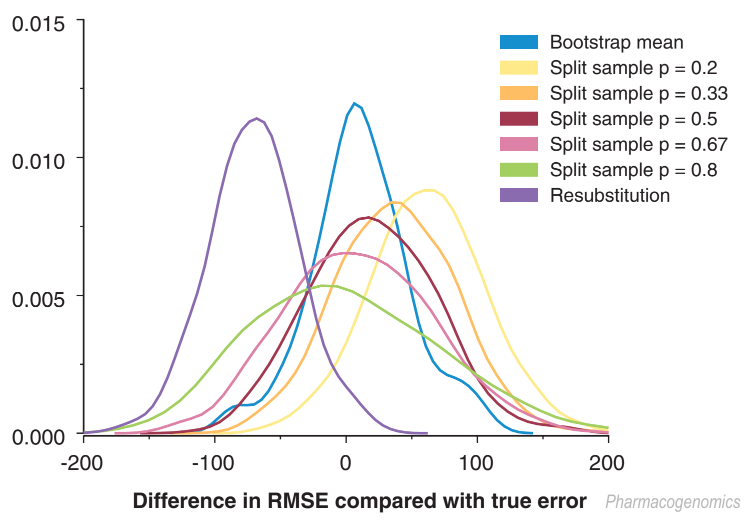
Performance estimation is a key, but often overlooked aspect of machine learning. A number of studies have described and compared the performance of different error estimators [7,8,10,11]; however, our own experience is that differences between competing approaches remain poorly understood in practical terms. Simplicity often seems to be the factor determining method choice, not only because simple methods are

Table 1. Comparison of different error estimators based on the proteasome data simulation.

Scheme	Bootstrap	Split sample					Resubstitution
		p = 0.2	p = 0.33	p = 0.5	p = 0.67	p = 0.8	
Bias	11	62	37	20	10	-0.3	-70
SD	25	33	35	39	49	64	21

The prediction error data on which the above statistics are based were calculated as root mean square error, predicted versus observed response. p is the proportion of training observations used by the split sample method. SD: Standard deviation.

Figure 2. Distributions for error estimators based on proteasome data.



The graphs plot the error difference $\mathrm{Err}_{True}(i) - \mathrm{Err}_X(i)$ where X is the given estimator and i is the ith sample. Densities were fitted using a kernel smoother with Gaussian weighting of neighboring points.
RMSE: Root mean square error.

easier to use, but also because researchers have greater confidence in results generated by methods in which inner workings are clear.

In the current study, we have attempted to untangle some of the complexity surrounding error estimation in a way that illustrates how various error estimators compare in real-world situations. Our findings were that the ever popular split sample approach is not optimal. In all cases studied, the bootstrap estimator had smaller variance and smaller bias (or nearly so). For binary response, bootstrap bias may be reduced still further by employing a bias correction such as 0.632+ [2]. We decided against this additional step for the current work because bootstrap bias was already low.

For any resampling approach, the user must decide upon the relative effort spent training prediction rules versus testing them, and to do this it is helpful to understand the consequences of this choice. As demonstrated with the finite difference calculation of equation (2), the rate of change of estimator variance experiences its sharpest increase when testing samples become small (for example, split sample designs with large *p*). On the other hand, using a small training sample increases bias, although the influence as measured by rate of change is less. These results are consistent with observed behavior of another kind of error estimator, leave-one-out cross-validation. Leave-one-out cross-validation uses the smallest possible test samples (single points), and although bias is almost zero, variance can be unacceptably high for binary response data [2,4].

Our conclusion that bootstrap is superior to the other estimators we examined is based on data from only two studies. Nonetheless, we feel the result is likely to be robust. First, the data from these studies were incorporated into Monte Carlo simulations rather than being analyzed as single samples. Second, the data themselves were quite different in terms of response, type of inputs and sample size. Finally, the qualitative behavior of the estimators for the two types of data was highly similar.

Results consistent with those in the current study have recently been reported in a study of binary response data, using small sample sizes (as small as 40 observations), a hybrid simulation design similar to the one in the current study, and where classification rules were generated by performing feature selection as a disjoint, preliminary step [8]. Bootstrap clearly outperformed a split sample approach, and was among the best of the range of estimators considered.

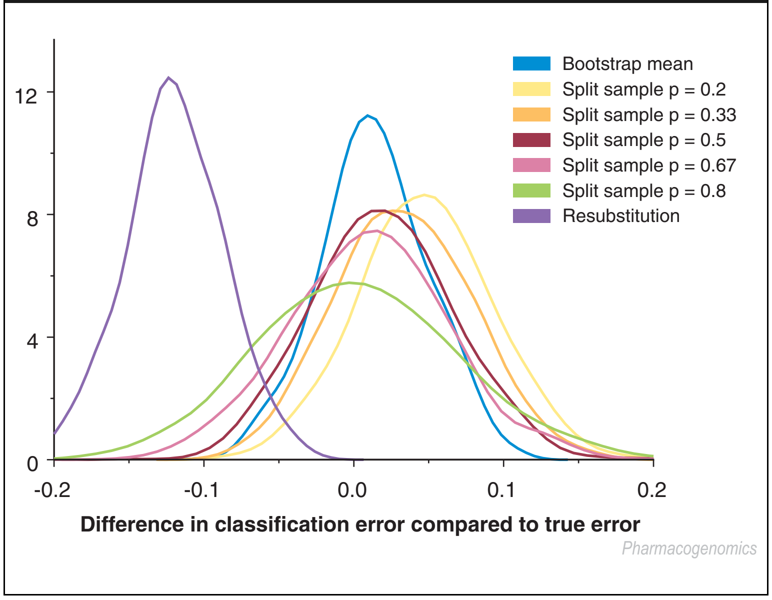
Other error estimators exist besides the ones chosen in the current study, in particular cross-validation approaches with or without repeated choice of the randomized partition (some are discussed in [11,12]). There are important connections between cross-validation and the sampling approaches used in the current study. For example, it has been shown that half-sample cross-validation with repeated randomized partition has strong

Table 2. Comparison of error estimators based on the antiplatelet data simulation.

Scheme	Bootstrap	Split sample					Resubstitution
		p = 0.2	p = 0.33	p = 0.5	p = 0.67	p = 0.8	
Bias	1.4	4.8	33	1.9	1.0	0.3	-12
SD	1.8	3.3	3.3	3.5	4.3	5.8	1.7

The classification error data on which the above statistics are based were calculated as the percentage of observations classified incorrectly; units are % error.
SD: Standard deviation.

Figure 3. Distributions for error estimators based on antiplatelet data.



similarities to the bootstrap [4], in terms of theoretical considerations, performance, and computational effort. Also, half-sample cross-validation using only a single randomized partition is similar to a split sample design with $p = 0.5$, the only difference being that cross-validation considers each half sample in turn as the training sample, while split sample chooses only one.

In a simulation-based comparison [2] of these and other error estimators in the context of binary response data, 0.632+ bootstrap performed best overall, with some forms of cross-validation also performing quite well. Split sample designs were not tested.

Based on this material, our expectation is the (*k*-fold) cross-validation with repeated randomized partition and small to medium *k* would perform similarly to bootstrap and out-perform the simpler split-sample approach. Note that our motivation for including split sample in the current study was not that we felt it to be the most powerful representative of available error estimators; rather, that in our experience, it is most commonly used.

A possible limitation of the current study is that results may depend, in part, upon the machine learner used: stochastic gradient boosting. Theoretically, gradient boosting is closely related to random forest [13]. Both use trees as a weak learner, and both generate an ensemble of resampled data using a bootstrap. Gradient boosting uses this ensemble to construct a functional expansion using gradient techniques, while random forest fits independent models to replicates in the ensemble and then aggregates them. Although random forest doesn't normally require an external estimate of error – since it instead uses the bootstrap error estimate formed automatically as the prediction rule is constructed – if an external estimate were required, we would expect the results of the current study to generalize to it. This is significant, because ensemble-based techniques such as gradient boosting and random forest have been shown to be among the most powerful machine learners available [6,7]. Whether the current results generalized beyond this is the basis for future research.

Error estimation is only useful because it helps to quantify our degree of belief in a final model, the model that we use for future prediction and for understanding a system's behavior. Such a final model is normally fitted to the entire baseline sample. However, some researchers employ a split-sample approach from the very beginning, reserving a portion of data solely for model testing. This is akin to strategies employed in some types of clinical trial and it gives maximal confidence that there is no 'cheating' involved. However, the price paid is high. When training of the final model is limited to a subset of the baseline data, model performance will be significantly worse than that of a model trained using the full, baseline data, a consequence of loss of sample size.

The results of this research have been incorporated into best practices for our own laboratories. For example, in a recent study of toxicogenomics [Martin R, Barros S. Manuscript submitted] we built classification rules to predict the toxic potential of drug candidates early in the process of pharmaceutical drug development. In order to implement these analytical findings as a live compound-screen, it has been essential to fully and precisely understand the accuracy of predictions made. We used bootstrap estimation to quantify the operating characteristics of our toxicity classifiers, in order to have maximum confidence that economic analyses that measured likely impact of such a screen, and which were dependent upon its performance, were accurate. This research is ongoing.

Outlook

The complexity and volume of genomics data is continuing to increase rapidly. For example, toxicogenomics studies in our own laboratories are currently based on microarray data with 9000 features. Next-generation microarrays are now available that increase feature availability to more than

30,000, and are projected to increase further in the future. An accompanying trend has been an increase in data complexity. A decade ago, genomics research was dominated by family-based linkage and association studies, whose data consisted of several hundred genetic markers plus a relatively modest amount of clinical measures. As research has shifted away from such positional cloning work, data for genomics studies have broadened to include gene expression and proteomics information, and new data constellations are likely to arise in the decade ahead. In our opinion, as these analytical challenges increase, so too will the need for robust assessment of results and conclusions.

Acknowledgments

The authors thank Victor Farutin for his expert help implementing stochastic gradient boosting. We also thank Archita D'Silva from MarketRx, Inc., for providing the antiplatelet survey data, and Tom Gant, Larry Dick, Chris Tsu, and Mark Rice for providing the proteasome inhibition data. Finally, we thank the reviewers for their constructive comments.

Highlights

- Machine-learning strategies are a standard approach for analyzing genomics data and for hypothesis generation.
- Confusion persists as to how to assess the prediction rules that result.
- Robust techniques are vital to quantify performance of the huge number of prediction rules being generated.
- Performance of prediction rules is normally assessed using their true error, but there are many choices for how this is estimated.
- Researchers favor simple error-estimation methods such as 'split sample' in which inner workings are clear.
- Four different error-estimation methods were compared using Monte Carlo simulation based on real data.
- In a variety of test situations, bootstrap was significantly more accurate than simpler methods such as split sample.
- Genomics researchers may be able to improve robustness of decision making by employing techniques such as bootstrap.

Bibliography

1. Breiman L: Statistical modeling: the two cultures. *Stat. Sci.* 16, 199–231 (2001).
2. Efron B, Tibshirani R: Improvements on cross-validation: the 0.632+ bootstrap method. *J. Am. Stat. Assoc.* 92, 548–560 (1997).
3. Hastie T, Tibshirani R, Friedman J: *The Elements of Statistical Learning.* Springer Verlag, New York, NY USA (2001).
4. Efron B: Estimating the error rate of a prediction rule. *J. Am. Stat. Assoc.* 78, 316–331 (1983).
5. Svetnik V, Wang T, Tong C, Liaw A, Sheridan R, Song Q: Boosting: an ensemble learning tool for compound classification and QSAR modeling. *J. Chem. Inf. Model.* 45, 786–799 (2005).
6. Wu B, Abbott T, Fishman D *et al.*: Comparison of statistical methods for classification of ovarian cancer using mass spectrometry data. *Bioinformatics* 19, 1636–1643 (2003).
7. Meyer D, Leisch F, Hornik K: The support vector machine under test. *Neurocomputing* 55, 169–186 (2003).
8. Molinaro A, Simon R, Pfeiffer R: Prediction error estimation: a comparison of resampling methods. *Bioinformatics* 21, 3301–3307 (2005).
9. Friedman J: Greedy function approximation: a gradient boosting machine. *Ann. Stat.* 29, 1189–1232 (1996).
10. Hawkins D, Basak S, Mills D: Assessing model fit by cross-validation. *J. Chem. Inf. Model.* 43, 579–586 (2002).
11. Efron B, Tibshirani R: *An Introduction to the Bootstrap.* Chapman & Hall, New York, NY, USA (1993).
12. Shao J: Linear model selection by cross-validation. *J. Am. Stat. Assoc.* 88, 486–494 (1993).
13. Breiman L: Random forests. *Machine Learning* 45, 5–32 (2001).